LIRA: Lightweight Incentivized Routing for Anonymity

Rob Jansen Aaron Johnson Paul Syverson

U.S. Naval Research Laboratory
{rob.g.jansen, aaron.m.johnson, paul.syverson}@nrl.navy.mil

***Abstract*—Tor, the most popular deployed distributed onion routing network, suffers from performance and scalability problems stemming from a lack of incentives for volunteers to contribute. Insufficient capacity limits scalability and harms the anonymity of its users. We introduce LIRA, a lightweight scheme that creates performance incentives for users to contribute bandwidth resources to the Tor network. LIRA uses a novel cryptographic lottery: winners may be guessed with tunable probability by any user or bought in exchange for resource contributions. The traffic of those winning the lottery is prioritized through Tor. The uncertainty of whether a buyer or a guesser is getting priority improves the anonymity of those purchasing winners, while the performance incentives encourage contribution. LIRA is more lightweight than prior reward schemes that pay for service and provides better anonymity than schemes that simply give priority to traffic originating from fast relays. We analyze LIRA's efficiency, anonymity, and incentives, present a prototype implementation, and describe experiments that show it indeed improves performance for those servicing the network.**

I. Introduction

Onion routing [1], particularly as deployed in Tor [2], [3] is the most widely used and extensively studied approach to anonymous communication. By protecting who is talking to whom and who is accessing which networks, Tor is a desirable tool for a variety of users. Its users include, but are not limited to: web users concerned about privacy; journalists, intelligence agents, and law enforcement agents concerned about hiding their operations; political activists concerned about surveillance from their government or from political opponents; and businesses concerned about industrial espionage or competitive intelligence.

Predominantly designed to provide anonymity for its users, Tor works by sending *client* traffic through multiple *relays* after encrypting it once for each relay in the *circuit*. Each relay decrypts and forwards traffic through the circuit as specified by the client. This traffic encryption and decryption process prevents a single member of the circuit from linking the user to its intended Internet destination. As it is paramount to Tor's design, anonymity has been improved by other design aspects: frequently rotating circuits and selecting the first relay from a small set of *guards* helps defend against passive logging attacks [4], [5] and further strengthens users' privacy.

Limited Capacity and Scalability. Tor relays are run by volunteers who altruistically contribute bandwidth and computational resources to the network. As a result, Tor is usable even by those unable or unwilling to contribute because they, e.g., have slow connections or are behind restrictive firewalls. Unfortunately, network capacity is limited to these altruistic contributions and has increased sublinearly to Tor's popularity. In Tor's current resource model, its popularity harms its usability and performance, and may therefore have a significant negative impact on its users' anonymity [6], [7]. The Tor Project [3] has enumerated many performance problems they have recognized and are actively pursuing designs that improve the network in this regard [8]. Recent work has focused on reducing the existing load on the network [9], [10] and optimizing utilization of the existing resources [11], [12], [13], but bolstering capacity while at the same time encouraging scalability remains a challenging open problem.

Various responses to this capacity and scalability problem have been considered. Thus far Tor has relied on community support to provide much-needed boosts to its capacity. For example, Torservers.net [14] is a registered German non-profit organization that uses donations to purchase or rent high-bandwidth servers for the public Tor network. Similarly, the Electronic Frontier Foundation ran a "Tor Challenge" in which they encouraged people to set up relays and listed the names of those who chose to be acknowledged for doing so [15]. Unfortunately, the support is limited and inadequate for Tor to scale to millions of simultaneous users while remaining usable. Currently Tor is initiating direct funding of relays using government funding it receives for this purpose. As noted in the blogpost announcement, this raises numerous questions, such as the impact on diversity of the infrastructure [16]. Another unknown is the sustainability of any resulting capacity increase if this direct funding ceases.

A more scalable way to increase capacity is to require all users to contribute in a peer-to-peer fashion. However, not only would it be difficult to force users to comply, this would also turn away some of those most in need of its protections due to an inability to contribute. Further, combined with a potential lack of user interest in operating and maintaining servers, this strategy may produce undesirable low bandwidth or unstable relays that increase network bottlenecks and may actually harm performance [17].

Numerous proposals to recruit new relays using incentives have appeared in the literature. Although the incentive approach is promising, past designs have thus far been plagued

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
FEB 2013		**00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
LIRA: Lightweight Incentivized Routing for Anonymity	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,4555 Overlook Ave., SW,Washington,DC,20375**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
20th Annual Network & Distributed System Security Symposium (NDSS '13), 24-27 Feb 2013, San Diego, CA.

14. ABSTRACT
Tor, the most popular deployed distributed onion routing network, suffers from performance and scalability problems stemming from a lack of incentives for volunteers to contribute. Insufficient capacity limits scalability and harms the anonymity of its users. We introduce LIRA, a lightweight scheme that creates performance incentives for users to contribute bandwidth resources to the Tor network. LIRA uses a novel cryptographic lottery: winners may be guessed with tunable probability by any user or bought in exchange for resource contributions. The traffic of those winning the lottery is prioritized through Tor. The uncertainty of whether a buyer or a guesser is getting priority improves the anonymity of those purchasing winners while the performance incentives encourage contribution. LIRA is more lightweight than prior reward schemes that pay for service and provides better anonymity than schemes that simply give priority to traffic originating from fast relays. We analyze LIRA?s efficiency, anonymity, and incentives, present a prototype implementation, and describe experiments that show it indeed improves performance for those servicing the network.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**17**	

with anonymity or efficiency problems. Both the "gold star" scheme [18] and Tortoise [9] have serious anonymity problems that allow relays' traffic to be identified, while PAR [19], XPAY [20], and BRAIDS [21] do not not scale well due to inefficient protocols. Our goal in this work is to design and evaluate a system that combines *strong anonymity* with *scalable efficiency*.

Lightweight Incentivized Routing for Anonymity. We present LIRA, a unique and scalable approach to creating incentives for users to contribute computational and bandwidth resources to Tor. Proportionally differentiated services [22] are the foundation for incentives: users who choose to run relays will be able to proportionally increase their performance relative to those not contributing. Further, relays may contribute more resources to increase the amount of their traffic that gets prioritized, leading to greater network capacity and performance improvements for everyone. At the same time, LIRA frustrates the adversary's ability to utilize traffic priority as a distinguisher of client-initiated and relay-initiated circuits.

LIRA produces incentives with a novel cryptographic *lottery* design together with a new circuit scheduling algorithm that prioritizes traffic from those winning the lottery. To play the relay lotteries, clients send a *ticket* to each relay in each circuit built in LIRA. Clients generate random number *guesses* to produce tickets locally, each of which will be a *winner* for a relay lottery with a tunable probability. Relays contributing resources may collect anonymous *coins* proportional to their contributions, and exchange the coins for guaranteed winners to relay lotteries of their choosing. Relays differentiate performance by prioritizing traffic for winning circuits.

LIRA maintains anonymity. An adversary in LIRA is unable to distinguish relays' purchased winners from clients' guessed winners, whereas an adversary in the "gold star" [18] and Tortoise [9] incentive designs can determine that traffic initiated from relays with absolute certainty. LIRA provides tunable anonymity: increasing the probability that a guessed ticket is a winner reduces the adversary's certainty about the traffic source.

LIRA is lightweight. Previous schemes either require that an online trusted third party participates in routing in order to prevent double spending, as in PAR [19] and XPAY [20], or that the third party distributes tickets to all relays *and all clients*, as in BRAIDS [21]. Neither of these approaches scales well to millions of simultaneous users. LIRA is scalable because purchased tickets are not managed for clients, but only for the orders of magnitude smaller set of relays, and there is no spending transaction when circuits are built.

Contributions. This work's major contributions may be summarized as follows:

- A unique and novel cryptographic lottery approach to providing incentives to run Tor relays that combines strong anonymity with scalable efficiency
- A new Tor circuit scheduler that produces performance incentives through proportional throughput differentiation
- A detailed efficiency, anonymity, and incentive analysis and comparison to BRAIDS [21], the state-of-the-art Tor incentive design
- A prototype implementation and experimental validation that LIRA provides incentives to contribute

Outline. The rest of the paper is outlined as follows. Section II provides details about the network, our threat model, and our objectives. LIRA's technical design is given in Section III, while Section IV analyzes LIRA's efficiency, anonymity, and incentives. Our protoype and experimental evaluation are described in Section V, Section VI discusses related work, and Section VII concludes.

II. Preliminaries

We now discuss specific details about the deployed Tor network that LIRA's design considers and describe the circuit building protocol to facilitate an understanding of how we will propose to modify it. We also introduce a bank as an additional service that will be utilized in LIRA's design, specify our adversarial threat model, and clarify the objectives of our system. Though LIRA could be applied to various anonymous communication systems, our exposition will focus on the Tor [2] onion routing network.

Onion-Routing Network. The most popular instantiation of onion routing [1], the Tor overlay network includes a *directory service* that publishes information about the available relays. Using the directory information, clients build three-hop circuits that begin with one of a small set of entry guard relays and end with an exit relay willing to connect to the client's desired Internet service. A circuit is built through a *telescoping* process: an encrypted tunnel is first created to an entry guard, after which the tunnel is extended one relay at a time until the circuit is completely established at the exit relay. During this building process, the client negotiates an ephemeral key with each relay in the circuit using a Diffie-Hellman key exchange protocol. Once established, client TCP streams that conform to the exit relay's exit policy may be multiplexed over the circuit for ten minutes, after which the circuit will be marked as dirty and will not permit any new application connections. The circuit is destroyed once existing application connections are done using it. All data transferred over the circuit is packaged into uniform-sized *cells* and encrypted using the negotiated ephemeral keys.

A relay may be servicing several circuits at any given time. Every circuit that results in data exchange between any pair of relays is multiplexed over a single TCP onion-routing connection between the pair. Cells read from this connection are processed and placed in a scheduling queue before being switched onto the corresponding outgoing onion-routing connection to the next-hop relay.

Roughly 3000 geographically diverse Tor relays currently transfer a combined total of about 1700 MiB/s from an estimated 400,000 unique users per day [23]. We parameterize the onion-routing network size for design and analysis purposes as m onion routers and n unique users in a given time period. We also assume the existence of a new bank service B. The bank will assist both in establishing valid lotteries with the relays and in assessing and rewarding contributions

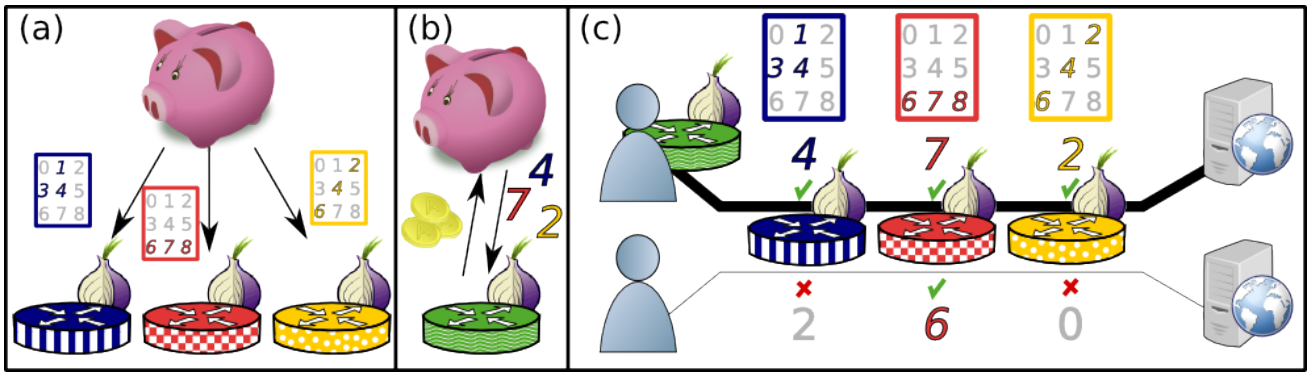


Fig. 1: An overview of LIRA's design. (a) Relays coordinate with the bank to learn which tickets will be winners for their lottery. (b) Relays accumulate anonymous coins by contributing bandwidth to Tor, and may exchange them for guaranteed winners to other relay lotteries. (c) Clients send either guaranteed winners or guesses through their circuits. Relays proportionally differentiate throughput by prioritizing circuits that submitted winners to every circuit position.

(see Section III). We will assume that all entities can use the underlying communication network to send each other messages directly.

Adversary. We will consider the actions of both a malicious network adversary and an honest-but-curious (*i.e.* passive) bank. We use the standard network adversary for onion routing [24], which is *local* in that he can observe part of the network, is *active* in that he can perform computations, send messages, run onion routers, and act as a client. Although in this paper we model the bank as a single entity, we expect the ultimate implementation of the bank to be similar to that of the directory service in the current public Tor network: multiple entities run the service and form a consensus on the authoritative documents. Therefore, we assume that the bank faithfully executes the protocol and only makes observations that are part of that protocol. In particular, he does not act as an onion router or as a client, only observes messages that are sent to him, and does not collude with the network adversary.

Objectives. Our goal is to provide incentive for anonymous-network users to run relays while preserving the desirable features of onion routing. Therefore, we will evaluate LIRA in terms of its functionality, its efficiency, the anonymity it provides, and the incentive it offers to run a relay.

We require that our system provide the functionality provided by onion routing. In particular, it should provide bidirectional, stream-oriented, low-latency communication between pairs of users. In addition, the responder of a stream should only need to run a standard transport protocol so users can communicate with destinations that aren't aware of or designed for anonymous communication protocols.

We also require that the efficiency of our system is comparable to onion routing. The success of Tor over alternative anonymous-communication protocols can be attributed in large part to its relatively low computational and communication costs. In particular, our protocol should have costs for each user that are proportional to amount of his anonymized traffic and for relays as a group that are proportional to the total amount of anonymized traffic. Moreover, we want the resource requirements at the bank to be achievable under current Tor network conditions and to scale well with a growing network.

Our evaluation will consider *relationship anonymity* [24] in our system, that is, the extent to which users can be linked to their communication partners. We will measure this using the probability that an adversary assigns to a user communicating with his actual destinations. We will also evaluate the incentives provided by LIRA. As it is designed to improve throughput and latency for users running relays, we will measure this performance difference while also considering the extent to which a user can cheat and obtain these improvements without contributing.

III. Design

To achieve the objectives stated in Section II, LIRA employs a cryptographic lottery and a relay circuit scheduler that prioritizes traffic for users who submit winning lottery tickets. A high level overview of LIRA's design and the interactions between these mechanisms is given in Figure 1. Through coordination with the bank, the relays receive information allowing them to recognize winning tickets to their own lottery (see Figure 1a). Over time, relays accumulate anonymous electronic coins from the bank by providing service to the network. These coins may be exchanged for guaranteed winning ticket values for a variety of relay lotteries (see Figure 1b). Clients without coins guess ticket values to produce them locally: their guesses will be winners with tunable probability. Tickets are passed to the relays through circuit control messages, and relays cannot distinguish a guessed winner from a guaranteed winner. Relays in every circuit position verify tickets and prioritize circuits of submitted winners by proportionally increasing their throughput (see Figure 1c). We now describe LIRA's design in further detail.

A. Setup

The bank will use RSA blind signatures [25], which allow it to sign a message without being able to link the signature with an earlier signing request. Let M be the RSA modulus of the bank, e be the public encryption exponent, and d be the corresponding private decryption exponent. Each relay r

will need a public random value $x_r \in \mathbf{Z}_M^*$ associated with it. These values can be generated by the bank and distributed by the directory service. For each relay r, the bank computes its signature x_r^d and sends it to r. Finally, the system will use a full-domain hash function $H : \{0,1\}^* \to \{0,1\}^{\lambda/2}$, where λ denotes the security parameter for the system. We can use a cryptographic hash function such as SHA-1 for H.

B. Coin Distribution

LIRA rewards relays proportional to the amount of bandwidth they contribute. Since relays can not be trusted to self-report their bandwidth contributions, we determine each relay's contribution with a secure bandwidth measurement scheme such as EigenSpeed [26]. Using EigenSpeed, relays opportunistically measure and evaluate each other's contributions to form an accurate consenus of relay bandwidth that has been shown to be resistant to attacks by malicious groups of colluding nodes [17], [26]. The measurement process runs continuously while a consensus is formed periodically.[1]

The bank stores and tracks each relay's bandwidth contribution over time, keeping an account balance of contributed bytes and updating it with each new bandwidth contribution consensus. A relay may collect ℓ digital *coins* from the bank for every α bytes it has contributed, where ℓ is the circuit length ($\ell = 3$ in Tor). A coin is constructed using a blind signature [25] to prevent the bank from later linking the coin to a given relay (the final signature is unknown to the bank).

Relays use their coins to purchase guaranteed winners from the bank (see Section III-C). The bank *prevents* double spending of these by keeping a database of previously spent coins that it checks (and possibly updates) when it receives coins in a purchase request. The size of the database is bounded by a coin expiration period η. A blind signature simplifies the construction of the coin over some electronic cash schemes since we do not require double spending *detection* by a third party after a coin has been successfully spent.

Advantages of using coins in the manner outlined above include flexibility and transferability. Coins are *flexible* because relays may accumulate them during periods when they are not actively using Tor as a client, and they are valid as long as the bank exists and the coin has not expired. The expiration period η is set so that the bank can store and access a list of spent, unexpired coins and may be adjusted as the Tor network scales. Section IV shows that currently in Tor 127.5 coins would be generated per second. If each coin is a 1024-bit signature, we can set $\eta = 28$ days, resulting in list of at most 4.60 GiB and fitting into a single machine's memory.

A coin is inherently *transferable* because it is not tied to a specific relay, allowing the possibility of a secondary economy to form around the purchase and sale of coins. In such an economy, it would also be possible for clients who do not run relays to obtain coins, improving anonymity by increasing the uncertainty of the sources of winning tickets.

[1]Tor currently computes the directory consensus every hour, which could be amended to include the bandwidth contribution information.

We configure the ratio of the number of contributed bytes α to the number of prioritized bytes β received in return to $\alpha = (\ell + 1) \cdot \beta$. By requiring a contribution $\ell + 1$ times that of prioritized consumption, we account for transferring data through each of the ℓ relays in the circuit, and also ensure that new relays that join Tor will only increase its overall capacity.

C. Purchasing Guaranteed Winners

Relays will prioritize traffic on circuits for which winning lottery tickets are supplied. Winners will be determined using a relay-specific permutation that we define below (Eq. 2). Let the size of the permutation's input space be 2^λ, and let $g_r : [2^\lambda] \to [2^\lambda]$ be the permutation of relay r. A value x is a winner for r if, for $y_0||y_1 = g_r(x)$, $y_0 \oplus y_1 < p2^{\lambda/2}$, where $p \in [0,1]$ is a system parameter. Thus a client that guesses an input x randomly will obtain a winner with probability p. To guarantee priority, a client can also use coins earned by providing service to the network to purchase winners.

Setting p presents a tradeoff between anonymity and incentives. Guessing a winner is less likely for smaller values of p. In this case, prioritized circuits are more likely to be paid for and thus probably originate at a client also running a relay. For larger values of p, it is more likely that a circuit will be prioritized by chance, and there will be less reason to run a relay and earn priority. (We discuss this tradeoff in more detail in Section IV.) We adopt a setting of $p = n^{-1/(2\ell)}$. For the current Tor network, we estimate n to be 10000, and thus we would set $p = 10^{-2/3} \approx 0.22$.

The construction of the permutations g_r is designed to provide properties similar to those of a pseudorandom permutation (PRP), although they are technically somewhat different. In particular, the permutations will appear sufficiently random to clients that they *cannot produce winners with probability significantly greater than p*. Moreover, they are *efficiently invertible* so that the bank can sell winners by choosing $y = y_0||y_1$ such that $y_0 \oplus y_1 < p2^{\lambda/2}$ and providing the corresponding input $g_r^{-1}(y)$. The construction also allows the purchase of winners for r to be made while *hiding the identity of r and the winning number from the bank*.

If we didn't want to hide this information from the bank, we could easily implement the rest of this functionality by using a PRP such as AES. The bank could share different private keys with each of the relays, and the user would simply purchase a winner by presenting a coin and specifying a relay. We wish to keep the bank as oblivious as possible, and thus we use a more involved construction for the lottery permutations.

The essential ingredient of the construction is for the bank to use blind signatures to obliviously provide a relay-specific input to a certain pseudorandom function (PRF) (Eq. 1). We then use a two-round version of the Luby-Rackoff construction [27] to convert the PRF into a permutation that is largely unpredictable to the relay.

1) Private Evaluation of Pseudorandom Functions: The PRF we use is adapted from one suggested by De Cristofaro et

1. c obtains blinded signature bx_r^d either from B or as protocol input.
2. c sends $bH(x)x_r^d$ to B.
3. B sends $H(H(x)x_r^d)$ to c.
4. c outputs $H(xH(H(x)x_r^d)$.

Fig. 2: PRF Protocol: c obtains $f_r(x)$ from B

al. [28] that can be computed obliviously.[2] Our construction doesn't provide full obliviousness with respect to the bank, but it will provide privacy assuming that the bank does not collude with a relay. The PRF used by relay r is

$$f_r(x) = H(x(H(H(x)x_r^d))), \tag{1}$$

where, as described above, $x_r \in \mathbf{Z}_M^*$ is publicly known.

The PRF Protocol for client c to obtain $f_r(x)$ from the bank B is given in Figure 2. We leave the option to obtain a blind signature in Step 1 as an input to the protocol to enable a batch-mode execution that will be used in the final purchase protocol. The client will be unable to guess outputs that he doesn't query with better than random chance because the relay signature x_r^d never appears in a message from the server that hasn't been blinded or hashed. Moreover, the protocol protects the privacy of the client's inputs (doing so is what prevents us from using a simpler PRF, such as $H(x_r^d x)$). In particular, the first unblinded input the bank sees has a factor $H(x)$ and thus appears random given that the bank doesn't know x. Including a factor of x before applying H in the last step prevents the bank itself from learning the final output.

2) Private Permutation Evaluation: Now we consider how to turn this into a permutation. Given $f : \{0,1\}^k \to \{0,1\}^k$, the *Feistel permutation* $D_f : \{0,1\}^{2k} \to \{0,1\}^{2k}$ on $x = x_1||x_0$, $x_1, x_0 \in \{0,1\}^k$, is defined as

$$D_f(x_1||x_0) = (x_0||x_1 \oplus f(x_0)).$$

This is invertible because x_0 is contained in the first k bits, and x_1 can be calculated as $f(x_0) \oplus (x_1 \oplus f(x_0))$. Luby and Rackoff showed that applying the Feistel permutation four times with four pseudorandom functions yields a pseudorandom permutation. We use this idea, but, in our setting, we will disallow winners at a relay that result in PRF inputs that have been used before. Thus, we can use the Luby-Rackoff construction with the single pseudorandom function f_r. In addition, we do not need the permutation output to appear random, but rather the XOR of the output halves. Thus, we are able to reduce the number of Feistel permutations to two. We therefore obtain a permutation for relay r of

$$g_r(x) = D_{f_r}\left(D_{f_r}\left(x\right)\right). \tag{2}$$

The permutation in Equation 2 is used by the relay to determine if a given ticket is a winner. To purchase a winner,

[2]The PRF they suggest is $H'(H(x)^d)$. To compute it, the client computes $H(x)$, obtains an RSA blind signature on it from the bank, and applies H' to the result.

1. c randomly chooses a and sends $a^e x_r$ to B.
2. B randomly chooses b and sends bax_r^d to c.
3. c uses the PRF Protocol with input bx_r^d to obtain $f_r(y_1)$ and sets $y_2 = f_r(y_1) \oplus y_0$.
4. c uses the PRF Protocol with input bx_r^d to obtain $f_r(y_2)$ and sets $y_3 = f_r(y_2) \oplus y_1$.
5. c outputs $y_3||y_2$.

Fig. 3: Permutation Protocol: c obtains $g_r^{-1}(y)$ from B

1. c pays B a digital coin.
2. c randomly chooses y_0, y_1 such that $y_0 \oplus y_1 < p2^{\lambda/2}$.
3. c uses the Permutation Protocol to obtain $g_r^{-1}(y_0||y_1)$.

Fig. 4: Winner Purchase Protocol: c purchases a winner for r from B

a client will actually choose a winning permutation output and apply the inverse permutation to obtain the ticket number. Let such an output be $y = y_1||y_0$, $y_0, y_1 \in \{0,1\}^{\lambda/2}$. The Permutation Protocol for a client c to obtain $g_r^{-1}(y)$ from the bank B is given in Figure 3. Observe that this protocol gives the client quite a bit more information about the function g^{-1} than a simple oracle query would. In fact, it reveals enough information for the client to determine values of g which he has not obtained via the protocol. However, this is only possible for a negligible quantity of inputs and we will show that relays can limit him to obtaining guaranteed priority only as many times as he has paid for winners (see Section IV).

3) Protocol to Purchase Winners: The Winner Purchase Protocol run by client c to purchase a winner for relay r from the bank B is given in Figure 4. We emphasize that the bank will only participate in step 3 of the Winner Purchase Protocol if c presents a valid coin. This protocol is run entirely over an anonymous onion-routing connection made by c to B. To prevent the bank from learning when encrypted circuits were made, clients should buy enough winners at a time to construct γ prioritized circuits, should maintain a reserve of enough winners to construct $\gamma/2$ prioritized circuits, and after depleting their reserve below this amount should wait a minimum time selected uniformly at random from $[0, \omega]$ before purchasing more winners.

We set γ to balance privacy with respect to the bank with the flexibility of the recommended buying strategy. Increasing γ increases the amount of time between batches of purchases observed by the bank and thus the number of other users' prioritized circuits that hide the buyer's circuits. On the other hand, decreasing γ decreases the number of coins a relay should have stockpiled before purchasing winners. We will ask a relay to run for at least 3 hours before purchasing winners. For a relay providing the current median bandwidth in Tor of 100 KiB/s [23] and with Tor's path length of $\ell = 3$, after 3 hours the relay obtains about 79 coins. Each prioritized circuit

1. c runs onion-routing circuit-creation protocol.
2. For each relay r in the circuit, c sends a TICKET cell with either a purchased winner w_r or random guess $x_r \in [1, 2^\lambda]$.
3. For every β bytes that pass over the circuit, c repeats Step 2 with new winners or guesses.

Fig. 5: Circuit Setup Protocol for client c

costs ℓ coins, and so we set $\gamma = 26$.

We set ω to maximize the number of prioritized connections that could have triggered a purchase without emptying the reserve of winners. Thus we set ω to the point at which a client's reserve of $\gamma/2$ winners would become empty had he been making only prioritized connections. For clients that make new circuits at rate r this happens at $\omega = \gamma/(2r)$. In Tor, $r \approx 1$, and so we have $\omega = 13$ minutes.

D. Circuit Setup

LIRA slightly modifies the onion-routing circuit-creation protocol (cf. Section II) to accommodate prioritization. Clients use a new TICKET cell type to send a lottery number to each relay on a circuit and attempt to obtain priority. A TICKET cell has the structure [TICKET, *number*], where the *number* field contains a value in $[0, 2^\lambda)$. This cell is sent to a relay from the client over the circuit and is thus onion-encrypted. In addition, relays on a circuit signal prioritization status to one another. These messages are sent directly over an encrypted pairwise connection (*e.g.* over the persistent TLS connections that Tor maintains between pairs of relays) with an identifier indicating to which circuit they pertain.

The Circuit Setup Protocol at the client is described in Figure 5. It simply adds periodic lottery-ticket messages to the standard circuit-creation protocol in onion routing. Note that the ticket messages are sent onion-encrypted over the circuit and thus can only be read by the recipient.

Relays determine priority for their circuits using the Circuit Priority Protocol described in Figure 6. For each position in a circuit, a relay maintains priority values for itself, for the preceding segment of the circuit, for the succeeding segment of the circuit, and for the entire circuit. The relay also maintains a counter for the number of priority bytes that are left under the current prioritization. Observe that the protocol involves explicit signaling along the circuit to synchronize the priority status of the relays. Thus, for the circuit depicted at the bottom of Figure 1c, even though the middle relay has received a winner, it will mark the circuit as DEAD after it receives a DEAD relay priority from either the guard or exit relay.

Also observe that, during the validation of a ticket number, the PRF inputs used during the two applications of the Feistel permutation (Eq. 2) are stored. If a PRF input used during ticket validation has been seen before, then that ticket is considered a loser. This prevents a client from reusing old winners and from using PRF outputs obtained from the bank to construct multiple winners. Finally, note that once a losing

Upon receiving data cell
  If priority bytes counter is greater than zero
    Reduce priority bytes counter by cell size
  If priority bytes counter is zero
    and self priority status is not DEAD
    Set self priority status to EXPIRED
    Set circuit priority status to EXPIRED.
Upon receiving priority status from adjacent relay
  Store status and send to other adjacent relay
  If all stored relay priorities are TRUE
    Set circuit priority TRUE
  If any stored relay priority is DEAD
    Set circuit priority DEAD
Upon receiving TICKET cell with value x
  Compute $y_0 || y_1 = g_r(x)$, storing intermediate PRF inputs
  If $y_0 \oplus y_1 < p2^{\lambda/2}$
    and intermediate PRF inputs previously unseen
    and no stored priority status is DEAD
    Set self priority status to TRUE
    Increment priority bytes counter by β
    Set circuit priority TRUE
  Else set self and circuit priority statuses to DEAD
  Send self priority status to adjacent relays

Fig. 6: Circuit Priority Protocol for relay r

ticket has been observed, the priority status is DEAD and no further priority is possible on the circuit.

The value of β, the number of bytes for which a winner provides priority, provides a tradeoff between user anonymity and the incentive to run a relay. A small β makes it unlikely that a *guesser*, that is, a user who does not buy winners, will maintain priority over the life of a typical circuit. This reduces the anonymity of a *buyer*, who we wish to allow to obtain priority for an entire connection. Setting a large β, assuming the price of a winning ticket increases proportionally, causes buyers to either use a circuit for a long time, reducing their anonymity by increasing the linkability of their connections, or to lose many of the bytes for which priority has been purchased, decreasing the incentive to earn it. In addition, a large β increases the granularity of buying winners, and so more e-cash must be earned before it can be used, again decreasing the incentive to earn e-cash.

Given these considerations, we use a value of β that is greater than the length of a typical connection. As discovered by McCoy et al. [29], over 90% of connections over Tor are HTTP connections. Ramachandran[30] shows data from billions of web pages showing that the mean size, including all embedded content, is 320KB. Cheng et al. [31] show that the mean YouTube file size is 8.4MB. To enable most web connections as well as such popular activities as viewing videos, we use $\beta = 10$MiB.

E. Circuit Scheduling

To improve the quality of service of qualifying traffic—cells on circuits for which valid winners have been provided—we incorporate ideas from the differentiated services architecture (DiffServ) [32]. More specifically, we ensure a relative quality ordering between qualifying and non-qualifying traffic using a priority scheduling mechanism based on the proportional differentiation model [22]. The model aims to provide *predictable* and *controllable* performance: quality metrics should be consistently proportional between classes and the proportions should be adjustable.

In the proportional differentiation model, traffic is separated into N classes labeled $c_1, \ldots, c_N$. The model states that the desired quality measurement q_i for each class c_i should be proportional to the other classes, where the proportions are configured with a differentation parameter p_i for each class. The classes should be scheduled such that the relative quality of each class follow the configured differentation parameters:

$$\forall i \in [N], \forall j \in [N] : \quad \frac{q_i(t, t+\sigma)}{q_j(t, t+\sigma)} = \frac{p_i}{p_j}$$

where $p_1 < p_2 < \ldots < p_N$, and σ is the measurement timescale. Dovrolis *et al.* explore *Proportional Delay Differentiation* and a priority scheduler that differentiates classes using queueing delay (packet waiting time) as the desired quality metric [33]. The scheduler utilizes two statistics to determine which class c_i to schedule at time t: the queueing delay $D_i(t)$ of the longest waiting packet in c_i, and the long-term average delay $\delta_i(t)$ of all previously scheduled packets (i.e., the average queueing delay of packets at the moment they are scheduled). The quality metric under Proportional Delay Differentiation becomes:

$$q_i(t) = D_i(t) \cdot f + \delta_i(t) \cdot (1 - f)$$

where f is an adjustable fraction (0.875 is suggested in [33]). A priority is computed for each c_i as $P_i(t) = q_i(t)/p_i(t)$, and the *longest* waiting packet from the class with the *maximum* computed priority is scheduled next.

Once a class is selected, the above approach is essentially first-come, first-served scheduling since each packet's delay timer starts when the packet enters the queue. However, it has been shown that an alternative approach is better suited to scheduling in the Tor network. In particular, prioritizing circuits with a low exponentially-weighted moving average (EWMA) circuit throughput may improve performance of bursty traffic while minimally harming bulk traffic with higher desired long-term throughput [11]. Therefore, LIRA schedules using *Proportional Throughput Differentiation*, where we adjust the quality metric at time t using the EWMA throughput of the lowest throughput circuit $T_i(t)$ and the long-term average throughput $\tau_i(t)$ of previously scheduled circuits:

$$q_i(t) = T_i(t) \cdot f + \tau_i(t) \cdot (1 - f)$$

where f remains adjustable. The priority is now computed for each c_i as $P_i(t) = q_i(t) \cdot p_i(t)$, and the circuit with the *lowest* EWMA throughput from the class with the *minimum* computed priority is scheduled next. Scheduling in this model allows us to configure the performance payoff associated with running a relay, or correctly guessing a winning ticket.

TABLE I: Bank Costs

Service	Operations	Messages
Coin generation	ρ signatures	ρ coin-size sent ρ coin-size received
Selling winners	ρf verifications $2\rho f$ hashes ρf signatures	ρf coin-size sent $2\rho f$ coin-size received $2\rho f$ winner-size sent $2\rho f$ winner-size received

IV. Analysis

A. Efficiency

LIRA preserves all the communication functionality of onion routing while providing both anonymity and efficiency. This section will consider how LIRA affects the overall computational and communication costs of the network.

Overhead. Clients may purchase a winner from the bank for each relay in a circuit to receive $\beta = 10$ MiB of prioritized traffic. Purchasing these winners involves ℓ RSA encryptions for the client portions of blind signatures and 4ℓ hashes. This cost is on the order of the cost for building a circuit, which is continuously incurred throughout a Tor client session. Clients that do not purchase winners incur no extra computational cost by using LIRA.

Our goal is to keep relay CPU costs low because, according to the Tor developers, the high-bandwidth Tor relays are CPU-bound. LIRA introduces some overhead for relays with ticket verification, *i.e.*, checking whether or not tickets are winners. This process involves evaluating the permutation in Equation 2 for every β bytes of transferred data. Each evaluation involves 6 hash computations, as well as a smaller number of multiplications and XORs. The DiffServ scheduler has been shown to be efficient [33] since each scheduling decision must only compute one priority for each class.

The bank is involved in distributing e-cash to relays and selling winning tickets. To generate e-cash, the bank creates a coin for every $\alpha/\ell = 10(\ell + 1)/\ell$ MiB sent by a given relay. Creating a coin involves a single blind signature, and these coins are given to the relays using a simple two-message protocol. To sell a winner, the bank must verify a coin by verifying a signature, provide a blind signature, and then participate in the batch PRF Protocol two times, each of which involves one hash.

We now consider the bank costs if the entire network is transferring b MiB/s and the fraction of coins that end up being used by the relays that earn them is f. Let $\rho = \ell b/(10(\ell + 1))$ be the rate at which coins are generated in this network. The rate of costly cryptographic operations and communicated messages for each bank service are outlined in Table I. It shows that the rate of the cryptographic operations is just a fraction of the total rate of traffic on the network. Table I also shows that the communication costs at the bank, in terms of the number of messages, the size of a digital coin, and the size of a ticket number, are similarly small.

Current Costs in Tor. To get a more concrete idea of what the costs at the bank might be in practice, we estimate what it would cost for a bank to serve the current Tor network. In Tor, $b = 1700$ and $\ell = 3$. The most costly operation by one to two orders of magnitude is signature generation. In the above setting, the rate of signature generation is $127.5+127.5f$ per second, where again $f \in [0, 1]$. OpenSSL benchmarks in Linux on an Intel Core2 Duo 2.67 GHz machine show that it is capable of creating 1705 1024-bit RSA signatures per second, and thus a modest machine is easily capable of generating the required signatures.

We also estimate the communication costs at the bank in this setting by using a signature size of 1024 bits and a ticket-number size of $\lambda = 320$ bits. Then the bank sends at a total rate of $15.94 + 25.90f$ KiB/s and receives at a total rate of $15.94 + 41.84f$ KiB/s, easily manageable with a single consumer-grade network connection.

Comparison to BRAIDS. To further understand LIRA's efficiency, we compare it to the efficiency of BRAIDS [21], the state-of-the-art Tor relay incentive design. The cost for each client in LIRA and BRAIDS are similar, however, only the clients that are purchasing guaranteed winners must pay this cost in LIRA as opposed to all clients that receive free tickets in BRAIDS. Further, a relay verifying winners in LIRA is at least an order of magnitude more efficient than a relay verifying tickets in BRAIDS: our OpenSSL benchmarks indicate LIRA's winner verification (6 SHA-1 hashes) takes roughly 18 microseconds, whereas a BRAIDS ticket verification takes roughly 1500 microseconds [21] on the same hardware.

The most important cryptographic cost is at the bank. LIRA's bank only needs to cryptographically pay the relays for some fraction of the total traffic due to its lottery design. On the other hand, the bank in BRAIDS pays for all traffic by distributing tickets to all clients. In addition, the number of ticket purchases is proportional to the amount of e-cash actually used by the relays to obtain prioritized service. If, as we expect, many relays altruistically provide more service than needed to support their own use, the system gains significant efficiency over distributing tickets to clients.

If we change the parameters of the BRAIDS scheme to more conservatively compare LIRA[3], we observe that BRAIDS requires at least 637.5 signatures per second. Even if $f = 1$ and relays spend all their credit, LIRA is more efficient. Moreover, BRAIDS requires a less computationally efficient partially-blind signature scheme. The signature-generation protocol in BRAIDS also has higher communication costs in both directions than RSA blind signatures, and thus is easily greater than LIRA's costs in both directions.

[3]We suppose that BRAIDS creates tickets for only half of the network traffic, as it is designed to cover Web traffic (58% of Tor traffic [29]). Also, we allow each ticket to buy 10MiB of priority, as in LIRA, and we give relays a bytes sent/earned ratio of 1/4, also as in LIRA. We then have (1700*3)/(2*4*10)=63.75 tickets created per second, which yields 63.75*20/2 = 637.5 total signatures per second when ticket exchanges are included.

B. Anonymity

We are interested in the extent to which the use of LIRA affects the anonymity of onion routing. Onion routing security is well-explored in the literature [24], [34], [35], [36], [37], and its vulnerabilities generally exist after adding our incentive system. Therefore, our goal is to prevent whatever anonymity is provided by onion routing from being significantly degraded. In this section we denote by $\mathsf{negl}(\lambda)$ a function that is negligible in λ [4].

Single Connection. Consider first a network adversary observing a single connection below the priority cutoff length of β. If the adversary is observing at a guard node, the user may be identified as running a relay if he is connecting to the guard from it. However, the multiple connections case shows that this would be quickly learned by the guard anyway. Thus it is a good idea for the user to use his own relays as guards.

Assuming the adversary is not observing at a guard node, from the adversary's perspective, LIRA simply adds TICKET cells and status signaling messages between pairs of relays. Users only directly affect the TICKET cells. Guessers and buyers generate these cells according to different distributions, potentially leading to some deanonymization. The difference lies in the probability that the TICKET cells contain a winner or not. Therefore, we need only consider an adversary's observations about whether or not the user inserts winning tickets into each of the relays on a circuit.

Suppose that a user does provide winners for an entire circuit. This happens with probability 1 for buyers and p^ℓ for guessers. The adversary can easily learn that submitted tickets were winners if he controls one of the circuit's relays. He may also learn this by observing the speed of traffic to and from a destination under his observation. Over time, the adversary can learn the distribution of traffic speeds for prioritized and unprioritized traffic and use the separation between these (as demonstrated in Section V) to infer the priority status of a given observed connection. Assume that buyers consist of the m relays and guessers consist of the other users of the network at a given time, and that each user is *a priori* equally-likely to create a circuit. Then the probability that the source of a connection is a given relay, based only on its circuit prioritization status, is $1/(m+(n-m)p^\ell)$, and the probability that it is a given non-relay is $p^\ell/(m+(n-m)p^\ell)$.

Now suppose that a user's tickets are not winners for the entire circuit. This happens with probability 0 for buyers and $1 - p^\ell$ for guessers. As before, the adversary can determine this in several ways. Buyers never fail to provide a winner, and so the adversary can infer that the source is a guesser. Given $n - m$ guessers, the probability that the source is a given one is $1/(n-m)$. (Of course buyers could intentionally fail to submit winning tickets at some relays periodically to complicate this analysis. We do not evaluate such possibility in this paper.)

We are most interested in the case that there are relatively

[4]A function f that is negligible in λ decreases faster with λ than any inverse polynomial. That is, $f(\lambda) = o(1/\lambda^k)$ for any k.

few buyers, as that would be true currently if LIRA were deployed in Tor, and we would expect it to remain so as long as the cost of running a relay is high relative to the benefit of anonymity for most users. In this case, the probability that a given buyer is the source of a single short prioritized connection, based only on its circuit prioritization status, is roughly $1/(m + np^\ell)$. With $p = n^{-1/(2\ell)}$, this becomes $1/(m+\sqrt{n})$. Thus, we can see that uncertainy over the source increases with the total number of users n, a desirable property of onion routing that we want to preserve. With few buyers, the probability that a given guesser is the source of a single short unprioritized connection, based only on the circuit's prioritization status, is roughly $1/n$, which is the best possible.

Of course, an adversary need not only take into account the prioritization status of a relay for purposes of deanonymization. Indeed, as discussed, all attacks on onion routing itself maybe be used in addition to the information provided by LIRA. However, the action of the incentive system is independent of the underlying onion routing protocol, and therefore the effect on deanonymization is simply to weight the distribution an adversary would otherwise infer. For example, suppose that, excluding the observations from the incentive system, the adversary can infer that the source is a given user with probability p_1. If that user has probability p_2 of achieving the priority observed, then, including those observations, the probability of the user becomes (proportional to) p_1p_2. One consequence of this as that LIRA increases the posterior probability of a buyer by at most $1/p^\ell$.

Multiple Connections. Circuits on which more than β bytes are sent include multiple TICKET cells from users. The above analysis applies to any one priority status, but taken together they degrade the anonymity of the user to the point that they are essentially identified as either a buyer or a guesser. Suppose a user's circuit transfers more than $(k-1)\beta$ bytes. This will happen when a single connection exceeds that amount. It can also happen if the total volume of multiple connections sent over the same circuit exceed that amount.[5] In this situation the user updates his priority status k times. The probability that a guesser maintains priority through all the updates is $p^{k\ell}$. Therefore, such a circuit created by a buyer quickly identifies him as a buyer, and the probability that it is a given buyer conditional only on the priorities observed is $1/(m + (n - m)p^{k\ell})$. Guessers are always identified as guessers when the tickets they submit fail to be winners, and this happens with an increasing likelihood of $1 - p^{k\ell}$.

Users making many circuits over time face the possibility of a similar decrease in anonymity. If an adversary can observe the priority status of k of a user's connection and link them together as belonging to the same (unknown) user, the resulting anonymity is just as if the user updated the priority status of a given circuit k times. Some ways the adversary may be able to link connections include controlling a destination at which users are active over long-lived sessions, controlling some exit nodes and linking together connection by related activities, and controlling some middle nodes and observing connections coming from the same guard nodes. The adversary is also be able to link connections at a guard node because they come from the source directly. A user running a relay may hope to hide that fact from a given guard by connecting to the anonymity network from a different location and buying tokens from a guard anonymously through a different guard. However, if he consistently buys priority, his guards will quickly determine that.

We note that hiding over the long term the fact that better service is being purchased seems to be a fundamental issue that any scheme will suffer from to some degree. In BRAIDS [21], for example, normal users receive fewer coins than relays, and so they can only be confused with relays if they save up many coins before buying, and thus few of them can buy at any one time. On the other hand, allowing users to purchase service without running a relay, which we ignored in our analysis due to uncertainty, has the potential to attract many more users than those that run relays. The Torservers.net project [14] already demonstrates that many prefer donating money to running relays. (Note that this also presents a mechanism whereby purchasing priority can indirectly add commensurate capacity to the network if all proceeds of such sales are directed into the purchase of more capacity, such as Torservers.net does.) In addition, the widespread use of VPNs for Internet security and blocking resistance indicate a willingness to pay for privacy.

Bank Privacy. We assume that the bank is semi-honest and only observes messages sent to it. The bank only observes the amount of e-cash earned by relays, when cash is transferred among users, and the purchase of winners. Clearly, then, the bank doesn't learn anything about the destinations of connection through the anonymity network, and therefore users have relationship anonymity with respect to the bank. However, LIRA protects user privacy even further.

First, all bank purchases are made using anonymous connections and anonymous coins. Therefore the bank doesn't learn who is spending e-cash and buying service in the network.

Second, clients batch and randomly time their purchase of ticket winners to hide when prioritized circuits are made from the bank. Clients should purchase $\gamma\ell$ winning tickets at a time. If a relay prioritizes all his circuits and makes them at Tor's rate $r \approx 1$ per minute, he purchases winners every $\gamma = 26$ minutes. Moreover, the time of the purchase triggered by a prioritization that reduces a client's reserve of winners below the $\gamma/2$ threshold is hidden from a bank within a period of $\omega = 13$ minutes. To get an idea of how many other prioritizations occur during these time periods, consider $n = 10000$ users making circuits at rate r, each gaining priority with probability $p^\ell = 1/\sqrt{n}$. Then during a 26 minute period between purchases there are an expected $\gamma np^\ell r = 2600$ prioritized circuits from other users and during a 13 minute period there are an expected 1300 such circuits.

Third, the Winner Purchase Protocol hides from the bank the relay identity and the ticket number of a purchased winner. The

[5]The amount of traffic sent over a circuit depends on the relative rates at which circuits and connection are created and destroyed. Tor only puts new connections on circuits that have been used for less than ten minutes, with a preference among used circuits for the youngest.

1. B sends challenge relays $r_0 \neq r_1$ to C.
2. C chooses a random bit $z \in \{0,1\}$.
3. C obtains a coin from B.
4. C executes the Winner Purchase Protocol with B for relay r_z.
5. B outputs a guess z' for the value of z.
6. The experiment value is 1 if $z' = z$, 0 if not.

Fig. 7: WPP-REL-IND experiment between B and C

indistinguishability experiment WPP-REL-IND between the bank B and a challenger C shown in Figure 7 tests how well the bank can determine the relay of a purchase. Theorem 1 shows that the observations a bank makes during a purchase for given relay are indistinguishable from the observations made during a purchase for a different relay.

Theorem 1: In the Random Oracle Model, $Pr[\text{WPP-REL-IND} = 1] \leq 1/2 + \text{negl}(\lambda)$.

Proof: Model H as a Random Oracle. Let Z represent the random bit chosen in Step 2 of the WPP-REL-IND experiment. We compare this experiment when $Z = 0$ and $Z = 1$ and show that, except with negligible probability, a given view of the adversary is equally likely in both cases. Let B make queries $Q_1, Q_2, \ldots, Q_t$ to H in that order during the experiment.

C begins the experiments for both $Z = 0$ and $Z = 1$ by paying B a coin. Coin generation and payment is independent of relay selection, and thus the observations B makes as part of those processes does not affect the probability of later observations and can be ignored.

Next, C chooses $O \leq Y_0, Y_1 < 2^{\lambda/2}$ such that $Y_0 \oplus Y_1 < p2^{\lambda/2}$. We can view C as choosing Y_1 independently and uniformly at random such that $O \leq Y_1 < 2^{\lambda/2}$. This implies that C chooses Y_0 in Step 2 independently and uniformly at random from the $p2^{\lambda/2}$ values that satisfy $0 \leq Y_0 \oplus Y_1 \leq p2^{\lambda/2}$. We let

$$\begin{aligned} Y_2 &= f_{r_Z}(Y_1) \\ &= Y_0 \oplus H(Y_1 H(H(Y_1)x_{r_Z}^d \end{aligned}$$

denote the value to be computes in Step 3 of the Permutation Protocol. Then we can define Collision to be the event that $Q_i = Y_1$ or $Q_i = Y_2$ for any $1 \leq i \leq t$.

C next sends $a^e x_{r_Z}$ to B to obtain the blinded signature $bx_{r_Z}^d$, where a is chosen independently and uniformly at random. As in the preceding, this observation can thus be ignored.

Next, C sends $X_1 = bH(Y_1)x_{r_Z}^d$ to B. Consider the queries $Q_1, \ldots, Q_{t_1}$ that occur before C queries $H(Y_1)$. Y_1 is independent of all observations of B at this point, and therefore the probability that $Q_i = Y_1$ for some $1 \leq i \leq t_1$ is $2^{-\lambda/2} \in \text{negl}(\lambda)$. Similarly, Y_0 is independent of all of B's observations, and thus the probability that $Q_i = Y_2$ for some $1 \leq i \leq t_1$ is $p2^{-\lambda/2} \in \text{negl}(\lambda)$. Assume from this point on that this doesn't happen.

The result of the query Y_1 to H is thus independently and uniformly random. Thus the probability of X_1 is the same whether $Z = 0$ or $Z = 1$. The former case, of course, implies that $H(Y_1) = X_1/(bx_{r_0}^d)$, while the latter implies that $H(Y_1) = X_1/(bx_{r_1}^d)$.

The next message from C to B is $X_2 = bH(Y_2)x_{r_Z}^d$. Let $Q_1, \ldots, Q_{t_2}$ be the queries that B makes to H up to the point that C queries Y_2. We have already assumed that $Q_i \notin \{Y_1, Y_2\}$ for $1 \leq i \leq t_1$.

■

We can define an experiment similar to WPP-REL-IND to test how well the bank can guess the ticket number of a purchase. It can be shown that the bank succeeds in that experiment with at most a negligible amount over a random guess as well.

C. Incentives

LIRA is designed to create an incentive for users to run relays or otherwise contribute to the system. We explore in Section V the extent to which it successfully provides better service to users that receive priority. Here we consider if users must, as we intend, earn e-cash in order to increase the amount of priority they can obtain. Again we denote by $\text{negl}(\lambda)$ a function that is negligible in λ.

We first note that the possibility of cheating the system is an important consideration but one less important than performance and preserving anonymity. Regardless of whether or not a user can deviate from the protocol to obtain more priority, the anonymity and performance properties of the system still hold. Thus system operators could experiment with the use of LIRA without compromising the properties the network already provides. Furthermore, the amount of cheating that will occur in practice in a network protocol is unclear. BitTorrent, for example, is susceptible to cheating [38], [39], but it tends to perform well in practice. Somewhat low barriers to cheating may well be sufficient to induce most participants to comply.

LIRA is designed to force users to pay in order to obtain a winner with probability greater than p. It achieves this by using an e-cash scheme and a novel cryptographic lottery.

In the e-cash scheme, users must present valid digital coins to participate in the Winner Purchase Protocol (Fig. 4). The e-cash scheme prevents coin forgery or double spending.

The winners themselves are obtained by participating in the Permutation Protocol (Fig. 3). This protocol allows users to observe much more about g_r than just the output. However, the intermediate PRF outputs that the users observe are only allowed by a relay to appear in one winner. If another ticket is submitted with a previously seen PRF value, the relay will treat it as a loser. Thus these intermediate values are of no use in producing more winners than were paid for, and on inputs with unseen intermediate PRF values, the XOR of the halves of the lottery permutation does indeed appear random.

We formalize this property in the security experiment PP between an adversary A and a challenger C shown in Figure 8. We will show that A succeeds in this experiment with at most a

1. A outputs relays $r = \{r_1, \ldots, r_k\}$ and a challenge relay $r_c \notin r$.
2. C outputs random values $\{x_{r_1}, \ldots, x_{r_k}, x_{r_c}\}$ in $\mathbf{Z}_M^*$ and signatures $\{x_{r_1}^d, \ldots, x_{r_k}^d\}$.
3. C executes the Permutation Protocol with A as many times t as requested.
4. A outputs $x = \{x_1, \ldots, x_{t+1}\}$.
5. If $\forall_i y_0^i \oplus y_1^i < p2^{\lambda/2}$, where $y_0^i || y_1^i = g_{r_c}(x_i)$ and no intermediate PRF input would be reused during protocol evaluations of the $g_{r_c}(x_i)$, the experiment value is 1; else it is 0.

Fig. 8: PP experiment between A and C

1. C generates RSA parameters (M, e, d) and outputs (M, e).
2. A outputs relays $r = \{r_1, \ldots, r_k\}$ and a challenge relay $r_c \notin r$.
4. C outputs random values $\{x_{r_1}, \ldots, x_{r_k}, x_{r_c}\}$ in $\mathbf{Z}_M^*$ and signatures $\{x_{r_1}^d, \ldots, x_{r_k}^d\}$.
5. C executes the PRF Protocol with A some t number of times.
6. A outputs $x = \{x_1, \ldots, x_t\}$, $\{y_1, \ldots, y_t\}$, $x_c \notin x$.
7. C randomly chooses $z \in \{0, 1\}$.
8. C outputs $f_{r_c}(x_c)$ if $z = 0$ and else a random y.
9. A outputs a guess z'. The experiment value is 1 if $z' = z$ and $\forall_i y_i = f_{r_c}(x_i)$. Otherwise it is 0.

Fig. 9: PRF experiment between A and C

negligible probability greater than p. But first, we show that the PRF Protocol actually provides the PRF properties. The PRF experiment shown in Figure 9 tests whether, after executing the PRF Protocol some arbitrary number of times t, an adversary A can distinguish $f_{r_c}(x)$ from a random value for more than t inputs x, where r_c is some challenge relay. Lemma 1 shows that the adversary succeeds in this experiment with probability at most a negligible amount over random chance.

Lemma 1: In the Random Oracle Model and under the RSA assumption, $Pr[\mathsf{PRF} = 1] \leq 1/2 + \mathsf{negl}(\lambda)$.

Proof: We can reduce winning this game to solving the RSA problem. We construct a simulator S for the PRF challenger C and random oracle H. S implements H internally. S implements parameter selection by relaying RSA parameters from the RSA challenger to PRF adversary A. S randomly selects the relay signatures $x_{r_i}^d$, computes the relay values x_{r_i} from them, and randomly selects x_{r_c}. S executes the challenger side of the PRF Protocol by storing the response values $w_i = H(x_i)$ output by A and using random values v_i for each $H(H(x_i)x_{r_c}^d)$.

Step 1 of the PRF Protocol is random, and so the distribution of these values given A's view is accurately produced by S. The value for $H(H(x_i)x_{r_c}^d)$ is indeed random in A's view unless the response from A in Step 2 of the PRF Protocol is such that $x_{r_c}^d w_i$ is equal to some value queried of H by A. S can check for this possibility by dividing all queries for H by each new w_i received from A, raising it to the power e, and comparing to x_{r_c}. If any verification succeeds, S submits that value to the RSA challenger. Under the RSA assumption, the probability that this happens is negligible. Assuming this doesn't happen, C randomly chooses a bit z and outputs a random y. If the outputs x_i and y_i of A are such that $H(x_i) = w_i$ and $y_i = H(v_i x_i)$, then S has not "programmed" H with a value for $H(x_c)x_{r_c}^d$ (i.e. choosing a value without knowing the input). S again checks if $H(x_c)x_{r_c}^d$ has been queried by dividing all queries by $H(x_c)$ and raising to the e, sending to the RSA challenger if so. Thus this happens with negligible probability. Assuming it hasn't, $f_{r_c}(x_c)$ is random from A's perspective, and the random y from C has the correct distribution. z is random and independent of y, and thus $z' = z$ with probability 1/2.

Therefore, overall C presents the correct view to A except with negligible probability, and thus we have $Pr[\mathsf{PRF} = 1] \leq 1/2 + \mathsf{negl}(\lambda)$. ■

Using Lemma 1, we can now show that the adversary does not have an advantage in finding more than t winners under the permutation g_r.

Theorem 2: In the Random Oracle Model, $Pr[\mathsf{PP} = 1] \leq p + \mathsf{negl}(\lambda)$.

Proof: After executing the Permutation Protocol t times, A only obtains the value of f_{r_c} on $2t$ inputs. Therefore, among the $2(t+1)$ values to which f_{r_c} is applied according to Equation 2, there must be at least one repeated value or one on which A has not evaluated f_{r_c}. If there is a repeated value, then the experiment value is 0. Otherwise, there is an unknown evaluation of f_{r_c} that appears random to A by Lemma 1. If this input to f_{r_c} appears as half of one of the x_i, the probability that its value under f_{r_c} results in a known input to f_{r_c} in the next Feistel permutation is thus negligible, and so one half of $g_{r_c}(x_i)$ appears random to A. If the unknown f_{r_c} input appears after the first Feistel permutation of some x_i, one half of $g_{r_c}(x_i)$ again appears random to A. In either case, the XOR of the halves of some $g_{r_c}(x_i)$ appears random to A. Thus that x_i is a winner with probability negligibly close to p. ■

While this theorem shows that users can only themselves produce unused winners with probability p, a user may try to game the network by creating circuits and determining their priority. If he attempts to do so without colluding with any relays, he must determine the priority of the circuit from its performance alone. Suppose that doing so requires that he send or receive at least c cells on a circuit. Then, to obtain a circuit with priority, the user must transfer an expected c/p total cells. If the cost of tranferring these cells is comparable to the amount of traffic a relay needs to transfer in order to earn enough coins to build a circuit, a rational user might choose to take that more-reliable option.

A user might also run or collude with a relay in order to obtain priority without paying. A relay on a circuit is able to determine from the messages between adjacent relays on a circuit its priority status. Therefore, a user could collude with a guard node to create and destroy circuits until one with

priority is obtained. Similarly, the user could collude with a middle or exit node, although given that the user in this case is presumably known to the colluding relay, it would seem only to improve performance without decreasing anonymity to directly connect to that relay. A simple remedy for this attack that renders it equivalent to testing and creating circuits is to defer the activation of priority on a circuit until some number c of cells have passed in either direction. The initial traffic on a circuit is fast even without priority due to EWMA scheduling, and so the performance impact should be minimal, although we have not implemented it in our experiments. An additional possible attack is for a colluding relay in the middle of the circuit to lie about the priority status of either side in order to get partial priority of a circuit. However, we again observe that it would seem to make little sense for a user to use a colluding relay as a middle node rather than connect directly. Also, for all attacks that involve a relay, the costs associated with running a relay are already being paid, and it would have to be the case that the cost of simply adding capacity is more than the cost of running a cheating scheme.

Finally, we observe that similar opportunities for cheating exist in other recent incentivization schemes for anonymous communication. The Tortoise scheme of Moore et al. [9] allows users to create many circuits to avoid throttling, similar to the multiple-circuits attacks in LIRA. Also, the BRAIDS design of Jansen et al. [21] is susceptible to malicious guards as well, in that guards can easily steal the winning tickets intended for their users. Thus while LIRA does not eliminate cheating, it does offer a substantially new balance among competing priorities.

V. Experiments

We simulate LIRA in an effort to understand the performance benefits possible when running our incentive scheme. Our experiments are done using Shadow [40], a scalable, high-fidelity network simulator that is capable of running real Tor binaries as plug-ins (using the available Shadow plug-in called Scallion [41]). Shadow allows us to create a private Tor network on a single machine and avoid privacy risks associated with live network experiments. Shadow experiments are completely controllable and repeatable, and are faithful to Tor's protocols since Shadow runs the real Tor software. In this section, we describe our configured experimental network environment, quantify its consistency with public Tor network performance, and explore how LIRA affects performance and improves incentives for a variety of users. Note that all of the experiments described in this section are repeated ten times to diminish random experimental variances, and each uses Tor software version `0.2.3.13-alpha`.

A. Network Model

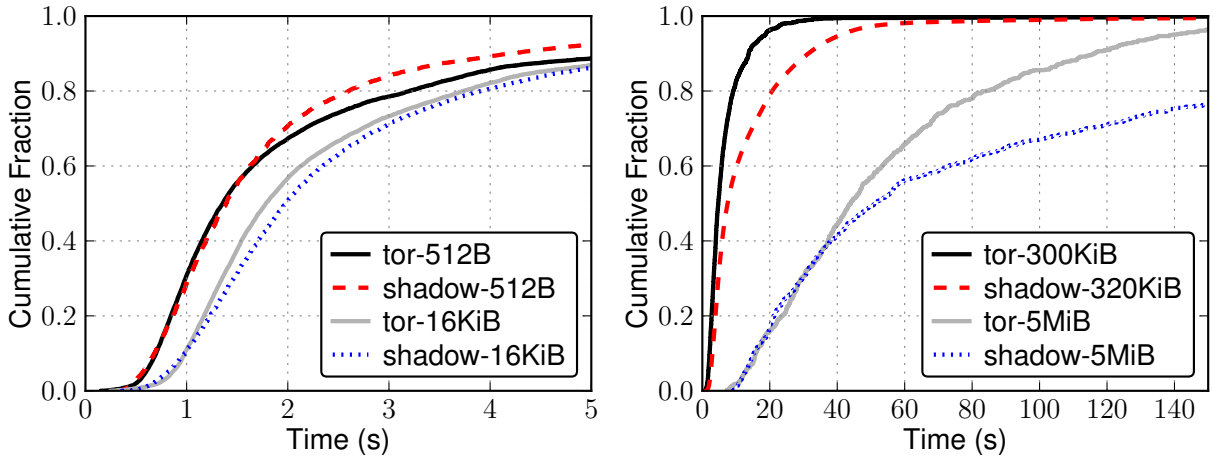
Shadow requires a complete-graph network topology that includes properties such as upstream and downstream bandwidths, latency, jitter, and packet loss. As network modeling is itself a challenging research problem, we rely on previous Tor network modeling contributions by Jansen *et al.* [42]. Their work considers every element of the Internet and the Tor network itself that must be modeled to run accurate Tor experiments in Shadow. Their model is built using real Internet measurements from GeoIP [43], iPlane [44], [45], and Net Index [46], and is validated with multiple experimentation platforms and data from the live Tor network itself [23].

We now give an overview of the Tor network model used in our experiments and discuss how it was modified from the original, the full details of which are presented in [42]. Our private Tor network consists of 50 generic `HTTP` servers, 50 Tor relays, 500 Tor clients. Of the 50 Tor relays, there is 1 directory authority, 20 exit relays, and 29 non-exit relays.

Although the original model configured 475 web and 25 bulk clients, a wider range of client applications would provide a more realistic traffic distribution. Therefore, we slightly modify the clients to better approximate Tor's protocol distribution as described in [29] and [47]. We configure 10 instant messaging clients (im), 465 web `HTTP` clients (web), 20 bulk `HTTP` clients (bulk), and 5 peer-to-peer clients (p2p). The im clients download 1 KiB files, pausing for one to five seconds after finishing one download and before starting the next. The web clients download 320 KiB files and pause for 1 to 20 seconds. The bulk clients continuously download 5 MiB files without pausing. All of the im, web, and bulk clients choose a random `HTTP` server for each download. The p2p clients form a "swarm" around a single 700 MiB file that is managed by a p2p authority. Each pair of p2p nodes connect and continuously exchange 16 KiB blocks of the file without pausing. Payload download times are measured as an indication of network performance.

Model and Simulation Accuracy. As we modified the model as originally described and validated in [42], we re-evaluate the consistency of Shadow's results with live network data. To determine how well our modeled network approximates client performance in the public Tor network, we compare download times in a vanilla Tor experiment with measurements of Tor collected by the TorPerf measurement system [48]. TorPerf downloads 50 KiB, 1 MiB, and 5 MiB files through the Tor network to monitor performance, and records various download times. Because our clients download differently sized files than TorPerf, we compare the time to receive the last byte of each of our experimental downloads with the time to receive the closest byte that is reported by TorPerf. As shown in Figure 10, Shadow does a reasonable job of characterizing the expected performance of the public Tor network. Performance for im and p2p clients are consistent with TorPerf measurements (Figure 10a), as are web and bulk downloads below approximately the fiftieth percentile (Figure 10b).

The difference in performance in the upper half of the distributions is possibly due to Tor's scheduling policy [11], in which circuit priority decreases as its throughput increases. TorPerf will have higher expected priority than clients in our experiments since TorPerf downloads once per circuit whereas our clients download multiple times per circuit. Note that we were unable to confirm this suspicion beyond reasonable doubt



(a) im and p2p clients

(b) web and bulk clients

Fig. 10: Shadow and public Tor performance are reasonably consistent for various transfer sizes.

due to a lack of experimental single file circuit downloads.

B. LIRA Prototype

We implement a research prototype of LIRA as described in Section III by directly modifying the Tor source code. To understand how to attribute changes in performance, we run separate experiments using the default EWMA circuit scheduling algorithm (vanilla Tor), our new Proportional Throughput Differentiation scheduler (diffserv) based on work by Dovrolis *et al.* [33] (see Section III-E), and various LIRA configurations (lira).

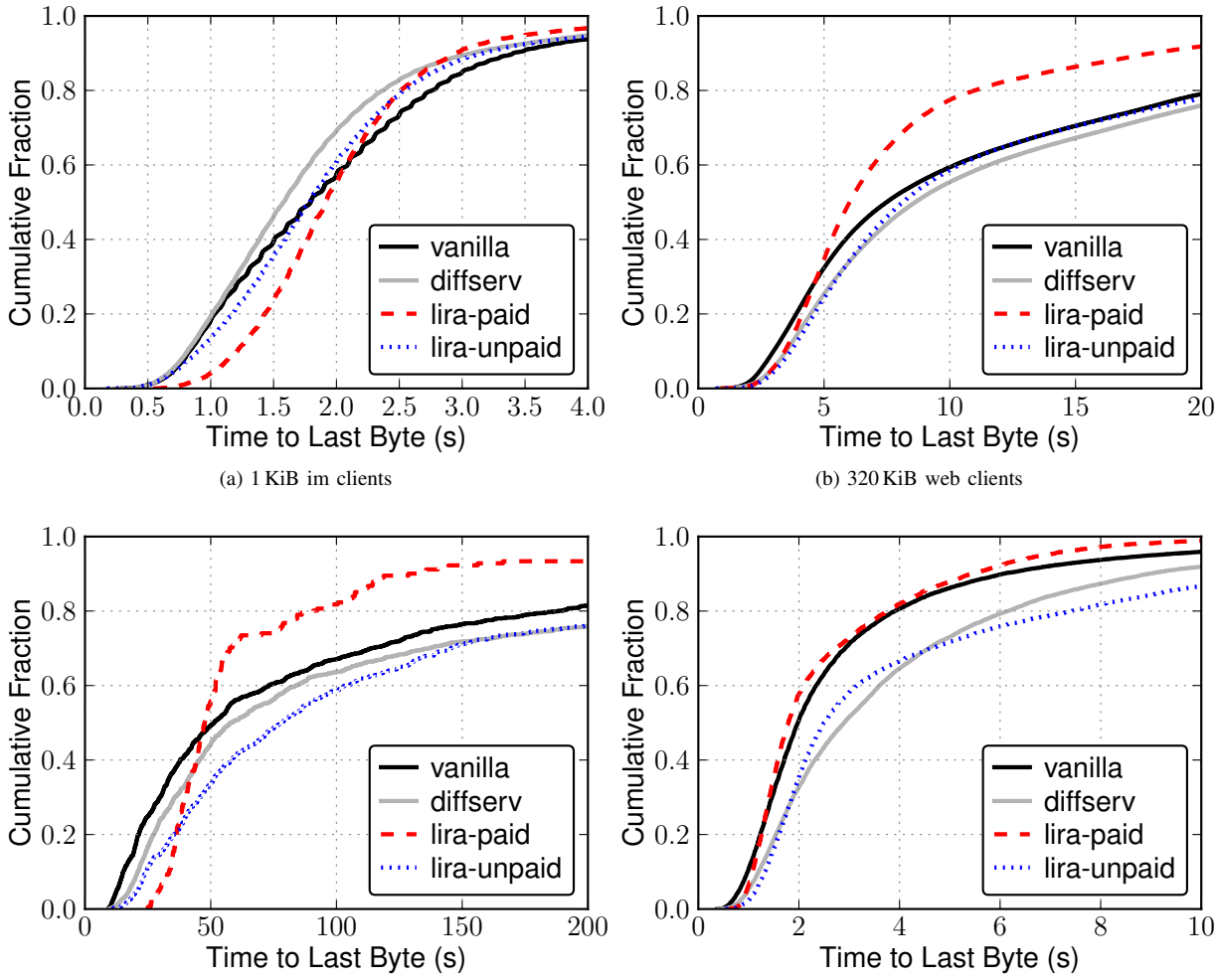
Class Differentation. We configure our new prototype scheduler with "paid" and "unpaid" classes c_1 and c_2, and differentiation parameters $p_1 = 1.0$ and $p_2 = 10.0$. Priorities are weighted by taking fraction $f = 0.875$ of the head-of-queue circuit EWMA, and 0.125 of the long-term class average EWMA. The EWMA throughput algorithms in both classes are configured with a 30 second half-life, which is also the default in our vanilla experiment and in public Tor. In our diffserv experiment, we isolate the new scheduler from the LIRA prototype: all clients are categorized in the unpaid class and there is no ticket guessing or buying. For each relay in our lira experiments, there is a corresponding client who uses that relay's winners to receive priority for all of its downloads. Of these 50 "paid" clients, we configure 1 im client, 47 web clients, 1 bulk client, and 1 p2p client. The remaining clients are "unpaid" and will only receive a prioritized circuit by correctly guessing with probability $p = 0.01$. Each prioritized circuit may be used for $\beta = 10$ MiB of data transfer, after which new guesses are submitted.

As shown by the cumulative distributions of download times in Figure 11, the new scheduler appears to give slightly preferential service to low throughput im clients and slightly worse to high throughput p2p clients. The scheduler tends to perform slightly worse than Tor's default scheduler, possibly because our prototype implementation has not been optimized. Our diffserv experiment provides a base upon which LIRA may be compared. The fundamental mechanism provided by the scheduler that is used to create performance incentives is tunable class differentiation. Figure 11 shows the scheduler's ability in this regard, as paid downloads are clearly differentiated from unpaid downloads. Note that the loss in performance for paid im downloads in Figure 11a is an artifact of the small sample (a single im client) and high latency due to unfavorable placement in the topology.

New Relay Capacity. Figure 11 shows performance in a network where only the existing Tor relays receive priority. We now explore a situation where several existing clients begin routing traffic for Tor. We consider networks where 5% and 15% of the existing client base[6] begin running a relay, adding a total of 25 and 75 relays and newly-paid clients to the existing sets of 50. Rationally, each new relay severely rate-limits its contribution so as to earn only enough winning tickets to support the expected throughput requirements of its client as computed from our vanilla experiment. This is a conservative estimate. Each client contributes four times its expected client throughput since LIRA will prioritize $1/4$ of a relay's contributed bytes when $\ell = 3$. Therefore, rate-limits are set to 20 KiB/s for those running im clients, 80 KiB/s for those running web clients, 340 KiB/s for those running bulk clients, and 128 KiB/s for those running p2p clients. Note that 1/3 of the added relays are exit relays, roughly the same proportion as in the current Tor network.

The rate-limiting outlined above results in 6.5% and 17.1% total additional network capacity, and represents a slightly pessimistic approximation of expected client contributions. To understand both extremes of the range of possible user behaviors, we configure other networks where the same 15% of new relays chosen above do not rate-limit their contributions. In the non-rate-limited networks, new relay bandwidth is sampled from the Net Index distribution [46] and results in a 95.7% and 383.5% increase in network capacity. Figure 12 shows

[6]Of the 5% of clients that begin running relays, we select 1 im, 23 web, 1 bulk, and 1 p2p. Of the 15%, we select 2 im, 69 web, 2 bulk, and 2 p2p.



(a) 1 KiB im clients

(b) 320 KiB web clients

(c) 5 MiB bulk clients

(d) 16 KiB p2p clients

Fig. 11: Client download time distribution in vanilla Tor, when using our proposed scheduler, and after adding LIRA's design modifications. LIRA adequately differentiates performance for paying clients without additional capacity.

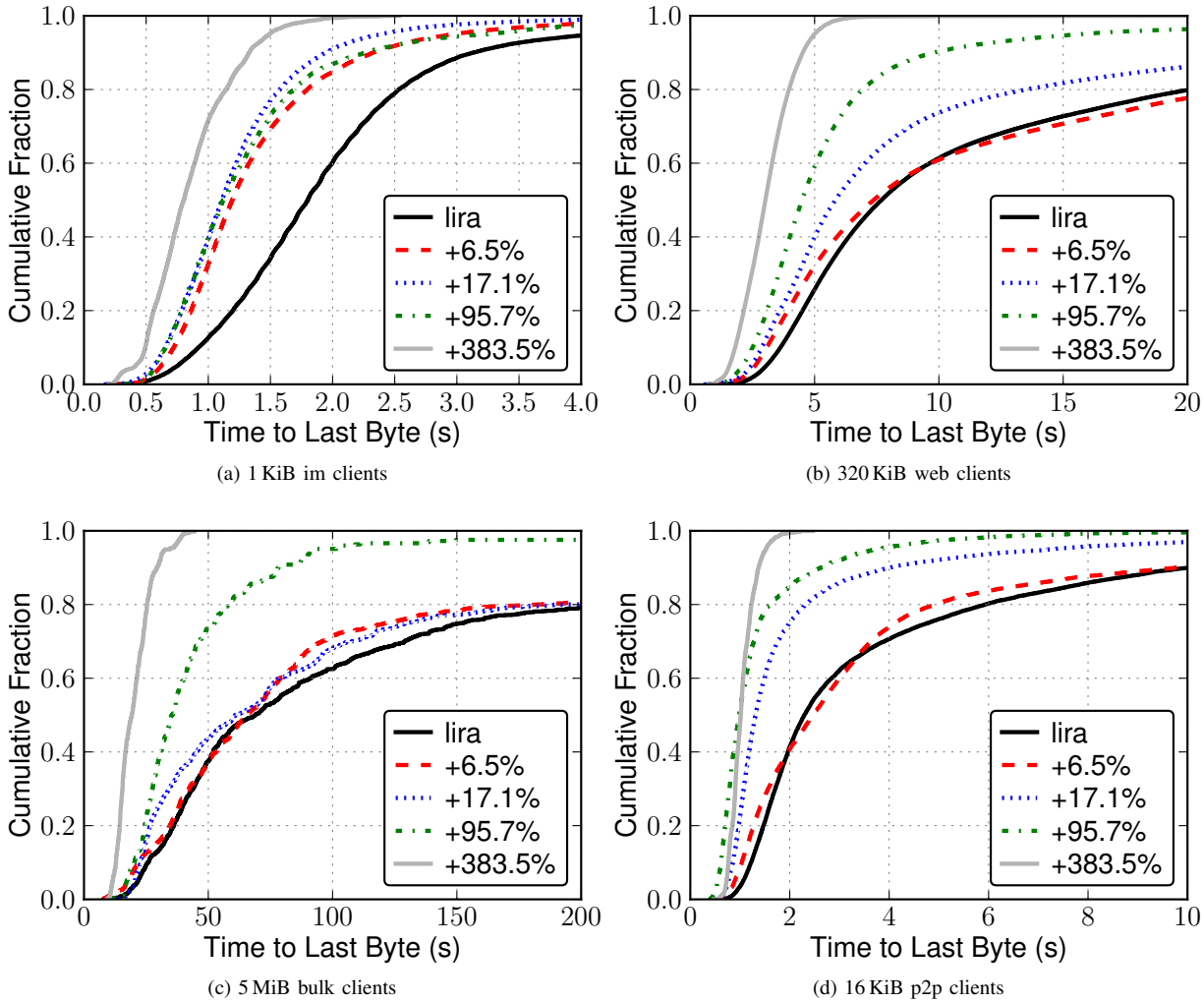
the result of additional capacity on client performance. As expected and not surprisingly, the added capacity results in a net increase in overall performance over LIRA without new relays, even under our rate-limiting scenarios. The net benefit to the network increases for all clients types, and more dramatically as more capacity is added. Our results confirm that LIRA (using the proportional throughput scheduler) enables performance incentives for contributors.

VI. Related Work

Several incentive schemes have been proposed for mix-nets [49], [50], [51] but do not directly apply to low-latency anonymous communication networks or Tor. Incentive designs for Tor include PAR [19], a scheme where relays accept real monetary payments from clients in return for routing service. PAR separates payments into anonymous coins paid by clients to guard relays, and more efficient identity-bound coins paid to the remaining relays. PAR and a similar micropayment scheme called XPAY [20] require an online bank to participate in the routing protocol to verify that the coins have not been double-spent. LIRA, however, is much more scalable as it does not require the bank to be involved in any routing transactions. LIRA also does not suffer from the fundamental trade-off between double-spending detection and accountability that plagues PAR and XPAY, wherein anonymity inherently decreases as the ability to detect cheaters improves.

Ngan *et al.* propose a lighter-weight scheme in which the fastest $7/8$ relays are marked with a "gold star" in the public Tor directory based on measurements by the directory servers [18]. These relays are given priority as they build circuits through other gold-star relays, and enjoy improved performance because only fast relays receive gold stars. Unfortunately, relay anonymity is reduced because the set of potential initiators of a prioritized circuit (the gold-star relays) is much smaller than that of an unprioritized circuit (any active client). LIRA manages the small anonymity set problem by allowing every user to receive priority by correctly guessing a winning ticket.

Jansen *et al.* propose BRAIDS [21], an incentive scheme that, similarly to LIRA, eliminates the double-spending prob-



(a) 1 KiB im clients

(b) 320 KiB web clients

(c) 5 MiB bulk clients

(d) 16 KiB p2p clients

Fig. 12: Client download time distribution as clients begin to run relays. Adding additional capacity as shown results in a net increase in performance considering both paid and unpaid clients.

lem by using relay-specific "tickets". Each relay may directly prevent the double spending of its tickets without needing to contact the centralized bank. BRAIDS aims to improve anonymity for relays by distributing tickets to all clients, using existing guard nodes as proxies for distribution. This strategy limits the total number of tickets that may exist in the system due to bank resource constraints and reduces payment flexibility. LIRA is more efficient in this regard since it must only manage ticket transactions for relays providing service—a much smaller set than that of all clients. BRAIDS also differentiates service [33] by asking clients to specify and pay for the desired class ("high throughput" or "low latency"), which may partition clients and negatively affect their anonymity. Conversely, clients do not choose their desired class in LIRA.

As an alternative to using e-cash or other payment-based cryptographic mechanisms to provide incentives, Moore *et al.* suggest in Tortoise [9] a universal rate limit of Tor clients and an exemption from such throttling for relays marked as `stable` and `fast` in the consensus. Unfortunately, this approach suffers from anonymity problems similar to the gold-star approach described above: the anonymity set of a given circuit is limited to a subset of relays, and the timing of relays' priority status appearing in the consensus leaks information that enables an intersection attack over time. LIRA, however, provides strong anonymity for well-defined spending levels. As in LIRA, Tortoise clients may multiplex traffic over multiple guards to evade throttling, thereby weakening the incentives provided by the system.

There has also been some work considering the behavior of participants in an anonymous-communication network from an economic perspective. Acquisti et al. [52] describe the costs and benefits of anonymity-network users, identify the challenges in designing systems that cope with selfishness, and present some possibilities for solving those challenges. Humbert et al. [53] provide an analysis of using a scrip system to incentivize selfish agents in a cooperative privacy-enhancing system such as an anonymity network. They establish the existence of a Nash equilibrium, examine its social welfare, and show how to manage the supply of scrip. Future study of

sociological behaviors in LIRA would be interesting should the system be deployed.

VII. Conclusion

The Tor network suffers from performance problems partially caused by a lack of relays willing to altruistically volunteer bandwidth. This paper presented LIRA, a novel incentive scheme that increases performance for those who contribute to the network by running a relay. We have shown that clients who choose to run relays enjoy faster downloads than those who don't, due to a novel ticket lottery design and a scheduler that differentiates service for winning tickets.

LIRA provides a higher degree of anonymity than previous proposals while eliminating the need for clients to contact the bank since, with tunable probability, clients can randomly self-produce winning tickets.

There are numerous directions for future work. Among them is developing a better understanding of the economics of anonymous incentives and how rational users might be expected to behave in LIRA or a similar design. Also useful would be a modified incentive structure that provides non-linear payoff for contributed capacity and higher payoff for more desirable relays such as bridges, exits, and those in more diverse geographic locations. A distributed bank that functions securely within Tor's trust model would improve scalability. Finally, better defenses against strategies for attempting to cheat the system and improved protection against long-term anonymity problems associated with linking paid high-throughput users would not only benefit LIRA, but any system attempting to provide anonymity-protected incentives.

VIII. Acknowledgements

We thank the anonymous reviewers for their feedback and suggestions for improving this work. This research was supported by the ONR and DARPA.

References

[1] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding routing information," in *Information Hiding: First International Workshop*, R. Anderson, Ed. Springer-Verlag, LNCS 1174, 1996, pp. 137–150.

[2] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*. USENIX Association, August 2004, pp. 303–319.

[3] "The Tor Project," https://www.torproject.org/.

[4] L. Overlier and P. Syverson, "Locating hidden servers," in *Security and Privacy, 2006 IEEE Symposium on*. IEEE, 2006, pp. 15–pp.

[5] M. Wright, M. Adler, B. N. Levine, and C. Shields, "Defending anonymous communication against passive logging attacks," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003, pp. 28–43.

[6] R. Dingledine and N. Mathewson, "Anonymity loves company: Usability and the network effect," in *Fifth Workshop on the Economics of Information Security (WEIS 2006)*, R. Anderson, Ed., June 2006.

[7] P. Syverson, "Why I'm not an entropist," in *Seventeenth International Workshop on Security Protocols*. Springer-Verlag, LNCS, 2009, forthcoming.

[8] R. Dingledine and S. Murdoch, "Performance improvements on tor or, why tor is slow and what were going to do about it," *Online: http://www.torproject. org/press/presskit/2009-03-11-performance. pdf*, 2009.

[9] W. B. Moore, C. Wacek, and M. Sherr, "Exploring the potential benefits of expanded rate limiting in tor: Slow and steady wins the race with tortoise," in *Proceedings of 2011 Annual Computer Security Applications Conference (ACSAC'11), Orlando, FL, USA*, December 2011.

[10] R. Jansen, P. Syverson, and N. Hopper, "Throttling tor bandwidth parasites," in *Proceedings of the 21st USENIX Security Symposium*. Internet Society, August 2012.

[11] C. Tang and I. Goldberg, "An improved algorithm for Tor circuit scheduling," in *CCS'10: Proceedings of the 13th ACM Conference on Computer and Communications Security*. ACM, 2010, pp. 329–339.

[12] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker, "Defenestrator: Throwing out windows in tor," in *Privacy Enhancing Technologies*. Springer, 2011, pp. 134–154.

[13] T. Wang, K. Bauer, C. Forero, and I. Goldberg, "Congestion-aware Path Selection for Tor," in *16th International Conference on Financial Cryptography and Data Security (FC)*, 2012.

[14] "Torservers.net," https://www.torservers.net/.

[15] "The EFF Tor Challenge," https://www.eff.org/torchallenge/.

[16] "Turning funding into more exit relays," https://blog.torproject.org/blog/turning-funding-more-exit-relays, July 2012.

[17] R. Snader and N. Borisov, "A tune-up for Tor: Improving security and performance in the Tor network," in *Proceedings of the Network and Distributed Security Symposium - NDSS '08*. Internet Society, February 2008.

[18] T.-W. J. Ngan, R. Dingledine, and D. S. Wallach, "Building incentives into Tor," in *Proceedings of Financial Cryptography (FC '10)*, R. Sion, Ed., January 2010.

[19] E. Androulaki, M. Raykova, S. Srivatsan, A. Stavrou, and S. M. Bellovin, "PAR: Payment for anonymous routing," in *Privacy Enhancing Technologies: 8th International Symposium, PETS 2008*, N. Borisov and I. Goldberg, Eds. Leuven, Belgium: Springer-Verlag, LNCS 5134, July 2008, pp. 219–236.

[20] Y. Chen, R. Sion, and B. Carbunar, "XPay: Practical anonymous payments for Tor routing and other networked services," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2009)*. ACM, November 2009.

[21] R. Jansen, N. Hopper, and Y. Kim, "Recruiting new Tor relays with BRAIDS," in *Proceedings of the 2010 ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, A. D. Keromytis and V. Shmatikov, Eds. ACM, 2010.

[22] C. Dovrolis and P. Ramanathan, "A case for relative differentiated services and the proportional differentiation model," *Network, IEEE*, vol. 13, no. 5, pp. 26–34, 1999.

[23] "Tor Metrics Portal," http://metrics.torproject.org/.

[24] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an analysis of onion routing security," in *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 96–114.

[25] D. Chaum, "Security without identification: transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, Oct. 1985. [Online]. Available: http://doi.acm.org/10.1145/4372.4373

[26] R. Snader and N. Borisov, "Eigenspeed: secure peer-to-peer bandwidth evaluation," in *Proceedings of the 8th international conference on Peer-to-peer systems*, ser. IPTPS'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 9–9. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855663.1855672

[27] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM J. Comput.*, vol. 17, no. 2, pp. 373–386, Apr. 1988. [Online]. Available: http://dx.doi.org/10.1137/0217022

[28] E. De Cristofaro, C. Soriente, G. Tsudik, and A. Williams, "Hummingbird: privacy at the time of twitter," in *33rd IEEE Symposium on Security and Privacy*. IEEE, 2012.

[29] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the Tor network," in *Privacy Enhancing Technologies: 8th International Symposium, PETS 2008*, N. Borisov and I. Goldberg, Eds. Leuven, Belgium: Springer-Verlag, LNCS 5134, July 2008, pp. 63–76.

[30] S. Ramachandran, "Web metrics: Size and number of resources," http://code.google.com/speed/articles/web-metrics.html, 2010.

[31] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *In the Proceeding of the 16th IEEE International Workshop on Quality of Service (IWQoS)*, 2008, pp. 229–238.

[32] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated service," United States, 1998.

[33] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: Delay differentiation and packet scheduling," *IEEE/ACM Transactions on Networking (TON)*, vol. 10, no. 1, pp. 12–26, 2002.

[34] J. Feigenbaum, A. Johnson, and P. Syverson, "Probabilistic analysis of onion routing in a black-box model," in *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society (WPES 2007)*, 2007, pp. 1–10.

[35] N. Evans, R. Dingledine, and C. Grothoff, "A practical congestion attack on tor using long paths," in *Proceedings of the 18th USENIX Security Symposium*, August 2009.

[36] N. Hopper, E. Y. Vasserman, and E. Chan-Tin, "How much anonymity does network latency leak?" *ACM Transactions on Information and System Security*, vol. 13, no. 2, February 2010.

[37] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz, "Denial of service or denial of security? How attacks on reliability can compromise anonymity," in *Proceedings of CCS 2007*, October 2007.

[38] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Do incentives build robustness in BitTorrent?" in *Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI 2007)*. USENIX, Apr. 2007. [Online]. Available: http://www.usenix.org/events/nsdi07/tech/piatek/piatek.pdf

[39] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer, "Free Riding in BitTorrent is Cheap," in *5th Workshop on Hot Topics in Networks (HotNets), Irvine, California, USA*, November 2006.

[40] R. Jansen and N. Hopper, "Shadow: Running Tor in a box for accurate and efficient experimentation," in *Proceedings of the Network and Distributed System Security Symposium - NDSS'12*. Internet Society, February 2012.

[41] "Shadow Code," http://github.com/shadow/.

[42] R. Jansen, K. Bauer, N. Hopper, and R. Dingledine, "Methodically Modeling the Tor Network," in *Proceedings of the 5th Workshop on Cyber Security Experimentation and Test*, August 2012.

[43] "MaxMind GeoIP," http://www.maxmind.com/.

[44] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An Information Plane for Distributed Services," in *Proc. of USENIX OSDI*, 2006, pp. 367–380.

[45] "iPlane: Datasets," http://iplane.cs.washington.edu/data/data.html.

[46] "Net Index Dataset," http://www.netindex.com/source-data/.

[47] A. Chaabane, P. Manils, and M. Kaafar, "Digging into anonymous traffic: A deep analysis of the tor anonymizing network," in *Network and System Security (NSS), 2010 4th International Conference on*. IEEE, 2010, pp. 167–174.

[48] "TorPerf measurement tools," https://gitweb.torproject.org/torperf.git/.

[49] E. Franz, A. Jerichow, and G. Wicke, "A payment scheme for mixes providing anonymity," *Trends in Distributed Systems for Electronic Commerce*, pp. 94–108, 1998.

[50] D. R. Figueiredo, J. K. Shapiro, and D. Towsley, "Using payments to promote cooperation in anonymity protocols," 2003.

[51] M. Reiter, X. Wang, and M. Wright, "Building reliable mix networks with fair exchange," in *Applied Cryptography and Network Security*. Springer, 2005, pp. 159–173.

[52] A. Acquisti, R. Dingledine, and P. Syverson, "On the economics of anonymity," in *Financial Cryptography. Springer-Verlag, LNCS 2742*, 2003, pp. 84–102.

[53] M. Humbert, M. Manshaei, and J.-P. Hubaux, "One-to-n scrip systems for cooperative privacy-enhancing technologies," in *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing*, 2011.